# Intel® Atom™ Processor N270 Series

## Specification Update

*July 2014*

*Revision 009*

Document Number: 320047-009US

# Contents

Preface ........ 5

Summary Tables of Changes ........ 7

Identification Information ........ 11

Errata ........ 13

Specification Changes ........ 29

Specification Clarifications ........ 30

Documentation Changes ........ 31

# Revision History

| Document Number | Revision | Description | Date |
|---|---|---|---|
| 320047 | 001 | • Initial Revision | June 2008 |
| 320047 | 002 | • Add Errata AAG32 and AAG33 | July 2008 |
| 320047 | 003 | • Add Errata AAG34 | September 2008 |
| 320047 | 004 | • Update Errata AAG34 | October 2008 |
| 320047 | 005 | • Update Errata AAG35 to AAG39<br>• Updated Specification Changes section | August 2009 |
| 320047 | 006 | • Added Errata AAG40 and AAG41 | January 2012 |
| 320047 | 007 | • Added Erratum AAG42 | March 2013 |
| 320047 | 008 | • Updated Erratum AAG_40 | May 2013 |
| 320047 | 009 | • Updated Note in Documentation Changes | July 2014 |

§

# Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

## Affected Documents

| Document Title | Document Number/Location |
|---|---|
| *Diamondville Processor EMTS, Revision 1.0* | Contact your Intel representative for the latest revision. |
| *RS- Diamondville Processor BIOS Writers Guide (BWG)* C | Contact your Intel representative for the latest revision. |

## Related Documents

| Document Title | Document Number/Location |
|---|---|
| *Intel® 64 and IA-32 Architectures Software Developer's Manual Documentation Changes* | 252046 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture* | 253665 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M* | 253666 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z* | 253667 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide* | 253668 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide* | 253669 |
| *IA-32 Intel® Architectures Optimization Reference Manual* | 248966 |
| *Intel® Processor Identification and the CPUID Instruction Application Note (AP-485)* | 241618 |

| Document Title | Document Number/Location |
|---|---|
| *Intel® 64 and IA-32 Architectures Application Note TLBs, Paging-Structure Caches, and Their Invalidation* | 317080 |

## Nomenclature

**Errata** are design defects or errors. These may cause the Intel® Diamondville Processor on 45-nm process behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

**S-Spec Number** is a five-digit code used to identify products. Products are differentiated by their unique characteristics, for example, core speed, L2 cache size, package type, etc. as described in the processor identification information table. Read all notes associated with each S-Spec number.

**QDF Number** is a four digit code used to distinguish between engineering samples. These samples are used for qualification and early design validation. The functionality of these parts can range from mechanical only to fully functional. This document has a processor identification information table that lists these QDF numbers and the corresponding product details.

**Specification Changes** are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

**Specification Clarifications** describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

**Documentation Changes** include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

***Note:*** Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

§

# Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

## Codes Used in Summary Table

### Stepping

| | |
|---|---|
| X: | Erratum, Specification Change or Clarification that applies to this stepping. |
| (No mark) or (Blank Box): | This erratum is fixed in listed stepping or specification change does not apply to list stepping. |

### Status

| | |
|---|---|
| Doc: | Document change or update that is implemented. |
| Plan Fix: | This erratum may be fixed in a future stepping of the product. |
| Fixed: | This erratum has been previously fixed. |
| No Fix: | There are no plans to fix this erratum. |

### Row

| | |
|---|---|
| Shaded: | This item is either new or modified from the previous version of the document. |

***Note:*** Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

| Number | Stepping | PLAN | ERRATA |
|---|---|---|---|
| | C0 | | |
| AAG1 | X | No Fix | A Write to an APIC Register Sometimes May Appear to Have Not Occurred |
| AAG2 | X | No Fix | An xTPR Update Transaction Cycle, if Enabled, May be Issued to the FSB after the Processor has Issued a Stop-Grant Special Cycle |
| AAG3 | X | No Fix | The Processor May Report a #TS Instead of a #GP Fault |
| AAG4 | X | No Fix | Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt |
| AAG5 | X | No Fix | MOV To/From Debug Registers Causes Debug Exception |
| AAG6 | X | No Fix | Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations |
| AAG7 | X | No Fix | Value for LBR/BTS/BTM will be Incorrect after an Exit from SMM |
| AAG8 | X | No Fix | Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update |
| AAG9 | X | No Fix | A Thermal Interrupt is Not Generated when the Current Temperature is Invalid |
| AAG10 | X | No Fix | Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts |
| AAG11 | X | No Fix | Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior |
| AAG12 | X | No Fix | Fault on ENTER Instruction May Result in Unexpected Value on Stack Frame |
| AAG13 | X | No Fix | With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction |
| AAG14 | X | No Fix | An Enabled Debug Breakpoint or Single Step Trap May Be Taken after MOV SS/POP SS Instruction if it is Followed by an Instruction That Signals a Floating Point Exception |
| AAG15 | X | No Fix | Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions and May Push the Wrong Address Onto the Stack |
| AAG16 | X | No Fix | BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer |
| AAG17 | X | No Fix | Single Step Interrupts with Floating Point Exception Pending May Be Mishandled |
| AAG18 | X | No Fix | Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results |
| AAG19 | X | No Fix | IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly |
| AAG20 | X | No Fix | Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set |
| AAG21 | X | No Fix | Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect |

| Number | Stepping | PLAN | ERRATA |
|---|---|---|---|
| | C0 | | |
| AAG22 | X | No Fix | Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts |
| AAG23 | X | No Fix | Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown |
| AAG24 | X | No Fix | IA32_MC1_STATUS MSR Bit[60] Does Not Reflect Machine Check Error Reporting Enable Correctly |
| AAG25 | X | No Fix | Split Locked Stores or Locked Stores Through Certain Segments May not Trigger the Monitoring Hardware |
| AAG26 | X | No Fix | When BIST is Enabled, Warm Reset Incorrectly Clears IA32_FEATURE_CONTROL MSR and the Last Exception Record MSRs |
| AAG27 | X | No Fix | CPUID Instruction Returns Incorrect Brand String |
| AAG28 | X | No Fix | The Instruction Cache Does Not Respond to Snoops When All Logical Processors on a Core Are in an Inactive State |
| AAG29 | X | No Fix | LINT0 Assertion and Deassertion During an Inactive State May Cause Unexpected Operation When APIC is Disabled |
| AAG30 | X | No Fix | Processor May Not Wake Up from an Inactive State When an Enhanced Intel® SpeedStep Technology Transition is Pending |
| AAG31 | X | No Fix | Thermal Interrupts are Dropped During and While Exiting Intel® Deep Power-Down State |
| AAG32 | X | No Fix | Corruption of CS Segment Register During RSM While Transitioning From Real Mode to Protected Mode |
| AAG33 | X | No Fix | TSC May be a Lower Value After Being in the Deeper Sleep State |
| AAG34 | x | No Fix | GP and Fixed Performance Monitoring Counters With AnyThread Bit Set May Not Accurately Count Only OS or Only USR Events |
| AAG35 | x | No Fix | CPUID Instruction Returns Incorrect Value For Leaf 0xA |
| AAG36 | x | No Fix | PMI Request is Not Generated on a Counter Overflow if Its OVF Bit is Already Set in IA32_PERF_GLOBAL_STATUS |
| AAG37 | x | No Fix | CPUID Indicates Wrong L2 Associativity in Leaf 80000006H |
| AAG38 | x | No Fix | Code Fetch May Occur to Incorrect Address After a Large Page is Split Into 4-Kbyte Pages |
| AAG39 | x | No Fix | Processor May Contain Incorrect Data and Hang Upon a Snoop When Combined with Specific Other Internal Conditions |
| AAG40 | X | No Fix | Complex Conditions Associated With Instruction Page Remapping or Self/Cross-Modifying Code Execution May Lead to Unpredictable System Behavior |
| AAG41 | X | No Fix | REP MOVS/STOS Executing With Fast Strings Enabled And Crossing Page Boundaries With Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations |
| AAG42 | X | No Fix | Paging Structure Entry May be Used Before Accessed And Dirty Flags Are Updated |

| Number | Stepping | PLAN | ERRATA |
|---|---|---|---|
| | C0 | | |
| **AAG1S** | X | No Fix | PEBS Record not Updated when in Probe Mode |
| **AAG2S** | X | No Fix | Microcode Update May Not Complete if any Logical Processor on a Core is in SMM |

| Number | SPECIFICATION CHANGES |
|---|---|
| AAG1 | Implementation of System Management Range Registers |

| Number | SPECIFICATION CLARIFICATIONS |
|---|---|
| AAG1 | Clarification of TRANSLATION LOOKASIDE BUFFERS (TLBS) Invalidation |

| Number | DOCUMENTATION CHANGES |
|---|---|
| 1 | Updated Note |

§

# Identification Information

Mobile Intel® Atom™ processor N270 series (code named Diamondville Processor-SC) on 45-nm process stepping can be identified by the following register contents:

**Table 1. Component Identification via Programming Interface**

| Reserved | Extended Family[1] | Extended Model[2] | Reserved | Processor Type[3] | Family Code[4] | Model Number[5] | Stepping ID[6] |
|---|---|---|---|---|---|---|---|
| 31:28 | 27:20 | 19:16 | 15:13 | 12 | 11:8 | 7:4 | 3:0 |
|  | 0000000b | 0001b |  | 0b | 0110b | 1100b | XXXXb |

When EAX is initialized to a value of 1, the CPUID instruction returns the Extended Family, Extended Model, Type, Family, Model and Stepping value in the EAX register. Note that the EDX processor signature value after reset is equivalent to the processor signature output value in the EAX register.

**NOTES:**

1. The Extended Family, bits [27:20] are used in conjunction with the Family Code, specified in bits [11:8], to indicate whether the processor belongs to the Intel386®, Intel486®, Pentium®, Pentium Pro, Pentium 4, or Intel Core processor family.
2. The Extended Model, bits [19:16] in conjunction with the Model Number, specified in bits [7:4], are used to identify the model of the processor within the processor's family.
3. The Processor Type, specified in bits [13:12] indicates whether the processor is an original OEM processor, an OverDrive processor, or a dual processor (capable of being used in a dual processor system).
4. The Family Code corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
5. The Model Number corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
6. The Stepping ID in bits [3:0] indicates the revision number of that model. See Table 2 for the processor stepping ID number in the CPUID information.

## Component Marking Information

Mobile Intel® Atom™ processor N270 series is identified by the following component markings.

**Figure 1. Mobile Intel® Atom™ Processor N270 Series (FCBGA8) Markings**

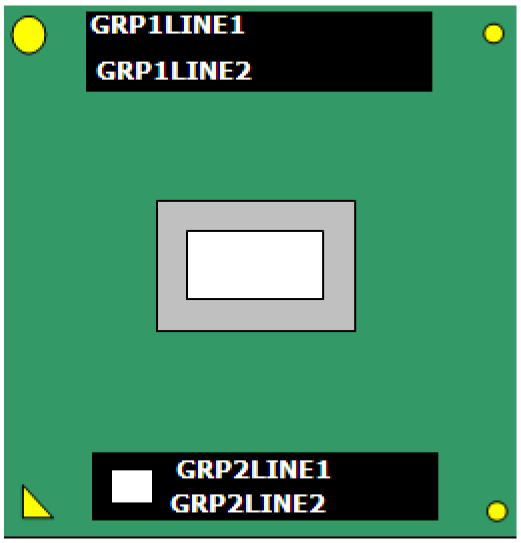

**Table 2. Identification Table for Mobile Intel® Atom™ Processor N270 Series**

| QDF/ S-spec | Product Stepping | HFM TDP (W) | Processor # | FSB Frequency | Processor Signature | Core Speed | | Package Micro-FCBGA-Pb= µ-BGA Lead Free | MCU |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Highest Freq. Mode (HFM) | Lowest Freq Mode (LFM) | | |
| QDTD | B0 | 2.5 | X | 533 MHz | 000106C1h | 1.6 GHz | 800 MHz | FCBGA8 | M01106C1109 |
| QDTB | B0 | 2.5 | X | 533 MHz | 000106C1h | 1.6 GHz | 800 MHz | FCBGA8 | M01106C1109 |
| QGZT | C0 | 2.5 | N270 | 533 MHz | 000106C2h | 1.6 GHz | 800 MHz | FCBGA8 | M04106C2218 |

§

# Errata

## AAG1 A Write to an APIC Register Sometimes May Appear to Have Not Occurred

**Problem:** With respect to the retirement of instructions, stores to the uncacheable memory based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, for example CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set, i.e. by STI instruction. Interrupts will remain pending and are not lost.

**Implication:** In this example the processor may allow interrupts to be accepted but may delay their service.

**Workaround:** This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG2 An xTPR Update Transaction Cycle, if Enabled, May be Issued to the FSB after the Processor has Issued a Stop-Grant Special Cycle

**Problem:** According to the FSB (Front Side Bus) protocol specification, no FSB cycles should be issued by the processor once a Stop-Grant special cycle has been issued to the bus. If xTPR update transactions are enabled by clearing the IA32_MISC_ENABLES[bit-23] at the time of Stop-Clock assertion, an xTPR update transaction cycle may be issued to the FSB after the processor has issued a Stop Grant Acknowledge transaction.

**Implication:** When this erratum occurs in systems using C-states C2 (Stop-Grant State) and higher the result could be a system hang.

**Workaround:** BIOS must leave the xTPR update transactions disabled (default).

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG3 Processor May Report a #TS Instead of a #GP Fault

**Problem:** A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

**Implication:** Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. Intel has not observed this erratum with any commercially available software.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG4 Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt

**Problem:** If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

**Implication:** An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector is left set in the in-service register and mask all interrupts at the same or lower priority.

**Workaround:** Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI; therefore the spurious vector should not be used when writing the LVT.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG5 MOV To/From Debug Registers Causes Debug Exception

**Problem:** When in V86 mode, if a MOV instruction is executed to/from a debug registers, a general-protection exception (#GP) should be generated. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

**Implication:** With debug-register protection enabled (i.e., the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception is generated instead of the expected general-protection fault.

**Workaround:** In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG6 Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations

**Problem:** An external A20M# pin if enabled forces address bit-20 to be masked (forced to zero) to emulates real-address mode address wraparound at 1 megabyte. However, if all of the following conditions are met, address bit-20 may not be masked.

- paging is enabled
- a linear address has bit-20 set
- the address references a large page
- A20M# is enabled

**Implication:** When A20M# is enabled and an address references a large page the resulting translated physical address may be incorrect. This erratum has not been observed with any commercially available operating system.

**Workaround:** Operating systems should not allow A20M# to be enabled if the masking of address bit-20 could be applied to an address that references a large page. A20M# is normally only used with the first megabyte of memory.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG7 Value for LBR/BTS/BTM will be Incorrect after an Exit from SMM

**Problem:** After a return from SMM (System Management Mode), the CPU will incorrectly update the LBR (Last Branch Record) and the BTS (Branch Trace Store), hence rendering their data invalid. The corresponding data if sent out as a BTM on the system bus will also be incorrect.
Note: This issue would only occur when one of the 3 above mentioned debug support facilities are used.

**Implication:** The value of the LBR, BTS, and BTM immediately after an RSM operation should not be used.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG8 Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update

**Problem:** A partial memory state save of the 512-byte FXSAVE image or a partial memory state restore of the FXRSTOR image may occur if a memory address exceeds the 64KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4GB limit while the processor is operating in 32-bit mode.

**Implication:** FXSAVE/FXRSTOR will incur a #GP fault due to the memory limit violation as expected but the memory state may be only partially saved or restored.

**Workaround:** Software should avoid memory accesses that wrap around the respective 16-bit and 32-bit mode memory limits.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG9 A Thermal Interrupt is Not Generated when the Current Temperature is Invalid

**Problem:** When the DTS (Digital Thermal Sensor) crosses one of its programmed thresholds it generates an interrupt and logs the event (IA32_THERM_STATUS MSR (019Ch) bits [9,7]). Due to this erratum, if the DTS reaches an invalid temperature (as indicated IA32_THERM_STATUS MSR bit[31]) it does not generate an interrupt even if one of the programmed thresholds is crossed and the corresponding log bits become set.

**Implication:** When the temperature reaches an invalid temperature the CPU does not generate a Thermal interrupt even if a programmed threshold is crossed.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG10 Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts

**Problem:** Software can enable DTS thermal interrupts by programming the thermal threshold and setting the respective thermal interrupt enable bit. When programming DTS value, the previous DTS threshold may be crossed. This will generate an unexpected thermal interrupt.

**Implication:** Software may observe an unexpected thermal interrupt occur after reprogramming the thermal threshold.

**Workaround:** In the ACPI/OS implement a workaround by temporarily disabling the DTS threshold interrupt before updating the DTS threshold value.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG11 Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior

**Problem:** Returning back from SMM mode into real mode while EFLAGS.VM is set in SMRAM may result in unpredictable system behavior.

**Implication:** If SMM software changes the value of the EFLAGS.VM in SMRAM, it may result in unpredictable system behavior. Intel has not observed this behavior in commercially available software.

**Workaround:** SMM software should not change the value of EFLAGS.VM in SMRAM.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG12 Fault on ENTER Instruction May Result in Unexpected Value on Stack Frame

**Problem:** The ENTER instruction is used to create a procedure stack frame. Due to this erratum, if execution of the ENTER instruction results in a fault, the dynamic storage area of the resultant stack frame may contain unexpected value (i.e. residual stack data as a result of processing the fault).

**Implication:** Data in the created stack frame may be altered following a fault on the ENTER instruction. Please refer to "Procedure Calls For Block-Structured Languages" in IA-32 Intel® Architecture Software Developer's Manual, Vol. 1, Basic Architecture, for information on the usage of the ENTER instructions. This erratum is not expected to occur in ring 3. Faults are usually processed in ring 0 and stack switch occurs when transferring to ring 0. Intel has not observed this erratum on any commercially available software.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG13 With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction

**Problem:** If an FP instruction generates an unmasked exception with the EFLAGS.TF=1, it is possible for external events to occur, including a transition to a lower power state. When resuming from the lower power state, it may be possible to take the single step trap before the execution of the original FP instruction completes.

**Implication:** A Single Step trap is taken when not expected.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG14 An Enabled Debug Breakpoint or Single Step Trap May Be Taken after MOV SS/POP SS Instruction if it is Followed by an Instruction That Signals a Floating Point Exception

**Problem:** A MOV SS/POP SS instruction should inhibit all interrupts including debug breakpoints until after execution of the following instruction. This is intended to allow the sequential execution of MOV SS/POP SS and MOV [r/e]SP, [r/e]BP instructions without having an invalid stack during interrupt handling. However, an enabled debug breakpoint or single step trap may be taken after MOV SS/POP SS if this instruction is followed by an instruction that signals a floating point exception rather than a MOV [r/e]SP, [r/e]BP instruction. This results in a debug exception being signaled on an unexpected instruction boundary since the MOV SS/POP SS and the following instruction should be executed atomically.

**Implication:** This can result in incorrect signaling of a debug exception and possibly a mismatched Stack Segment and Stack Pointer. If MOV SS/POP SS is not followed by a MOV [r/e]SP, [r/e]BP, there may be a mismatched Stack Segment and Stack Pointer on any exception. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** As recommended in the IA32 Intel® Architecture Software Developer's Manual, the use of MOV SS/POP SS in conjunction with MOV [r/e]SP, [r/e]BP will avoid the failure since the MOV [r/e]SP, [r/e]BP will not generate a floating point exception. Developers of debug tools should be aware of the potential incorrect debug event signaling created by this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG15 Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions and May Push the Wrong Address Onto the Stack

**Problem:** Normally, when the processor encounters a Segment Limit or Canonical Fault due to code execution, a #GP (General Protection Exception) fault is generated after all higher priority Interrupts and exceptions are serviced. Due to this erratum, if RSM (Resume from System Management Mode) returns to execution flow that results in a Code Segment Limit or Canonical Fault, the #GP fault may be serviced before a higher priority Interrupt or Exception (for example NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc.). If the RSM attempts to return to a non-canonical address, the address pushed onto the stack for this #GP fault may not match the non-canonical address that caused the fault.

**Implication:** Operating systems may observe a #GP fault being serviced before higher priority Interrupts and Exceptions. Intel has not observed this erratum on any commercially available software.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG16 BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer

**Problem:** If the BTS/PEBS buffer is defined such that:

- The difference between BTS/PEBS buffer base and BTS/PEBS absolute maximum is not an integer multiple of the corresponding record sizes
- BTS/PEBS absolute maximum is less than a record size from the end of the virtual address space
- The record that would cross BTS/PEBS absolute maximum will also continue past the end of the virtual address space

A BTS/PEBS record can be written that will wrap at the 4G boundary (IA32) or 2^64 boundary (EM64T mode), and write memory outside of the BTS/PEBS buffer.

**Implication:** Software that uses BTS/PEBS near the 4G boundary (IA32) or 2^64 boundary (EM64T mode), and defines the buffer such that it does not hold an integer multiple of records can update memory outside the BTS/PEBS buffer.

**Workaround:** Define BTS/PEBS buffer such that BTS/PEBS absolute maximum minus BTS/PEBS buffer base is integer multiple of the corresponding record sizes as recommended in the IA-32 Intel® Architecture Software Developer's Manual, Volume 3.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG17 Single Step Interrupts with Floating Point Exception Pending May Be Mishandled

**Problem:** In certain circumstances, when a floating point exception (#MF) is pending during single-step execution, processing of the single-step debug exception (#DB) may be mishandled.

**Implication:** When this erratum occurs, #DB is incorrectly handled as follows:

- #DB is signaled before the pending higher priority #MF (Interrupt 16)
- #DB is generated twice on the same instruction

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG18 Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results

**Problem:** The act of one processor, or system bus master, writing data into a currently executing code segment of a second processor with the intent of having the second processor execute that data as code is called cross-modifying code (XMC). XMC that does not force the second processor to execute a synchronizing instruction, prior to execution of the new code, is called unsynchronized XMC. Software using unsynchronized XMC to modify the instruction byte stream of a processor can see unexpected or unpredictable execution behavior from the processor that is executing the modified code.

**Implication:** In this case, the phrase "unexpected or unpredictable execution behavior" encompasses the generation of most of the exceptions listed in the Intel Architecture Software Developer's Manual Volume 3A: System Programming Guide, including a General Protection Fault (#GP) or other unexpected behaviors.

**Workaround:** In order to avoid this erratum, programmers should use the XMC synchronization algorithm as detailed in the *Intel Architecture Software Developer's Manual Volume 3A: System Programming Guide*, Section: Handling Self- and Cross-Modifying Code.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG19 IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly

**Problem:** The IO_SMI bit in SMRAM's location 7FA4H is set to "1" by the CPU to indicate a System Management Interrupt (SMI) occurred as the result of executing an instruction that reads from an I/O port. Due to this erratum, the IO_SMI bit may be incorrectly set by:

- A SMI that is pending while a lower priority event is executing
- A REP I/O read
- A I/O read that redirects to MWAIT

**Implication:** SMM handlers may get false IO_SMI indication.

**Workaround:** The SMM handler has to evaluate the saved context to determine if the SMI was triggered by an instruction that read from an I/O port. The SMM handler must not restart an I/O instruction if the platform has not been configured to generate a synchronous SMI for the recorded I/O port address.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG20 Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set

**Problem:** When the FREEZE_LBRS_ON_PMI, IA32_DEBUGCTL MSR (1D9H) bit [11], is set, future writes to IA32_DEBUGCTL MSR may not occur in certain rare corner cases. Writes to this register by software or during certain processor operations are affected.

**Implication:** Under certain circumstances, the IA32_DEBUGCTL MSR value may not be updated properly and will retain the old value. Intel has not observed this erratum with any commercially available software.

**Workaround:** Do not set the FREEZE_LBRS_ON_PMI bit of IA32_DEBUGCTL MSR.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG21 Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect

**Problem:** When an L2 Cache error occurs (Error code 0x010A or 0x110A reported in IA32_MCi_STATUS MSR bits [15:0]), the address is logged in the MCA address register (IA32_MCi_ADDR MSR). Under some scenarios, the address reported may be incorrect.

**Implication:** Software should not rely on the value reported in IA32_MCi_ADDR MSR for L2 Cache errors.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG22 Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts

**Problem:** Interrupts that are pending prior to the execution of the STI (Set Interrupt Flag) instruction are normally serviced immediately after the instruction following the STI. An exception to this is if the following instruction triggers a #MF. In this situation, the interrupt should be serviced before the #MF. Because of this erratum, if following STI, an instruction that triggers a #MF is executed while STPCLK#, Enhanced Intel Enhanced Intel SpeedStep® Technology transitions or Thermal Monitor events occur, the pending #MF may be serviced before higher priority interrupts.

**Implication:** Software may observe #MF being serviced before higher priority interrupts.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG23 Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown

**Problem:** According to the Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3A, “Exception and Interrupt Reference”, if another exception occurs while attempting to call the double-fault handler, the processor enters shutdown mode. Due to this erratum, any benign faults while attempting to call double-fault handler will not cause a shutdown. However Contributory Exceptions and Page Faults will continue to cause a triple fault shutdown.

**Implication:** If a benign exception occurs while attempting to call the double-fault handler, the processor may hang or may handle the benign exception. Intel has not observed this erratum with any commercially available software.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG24 IA32_MC1_STATUS MSR Bit [60] Does Not Reflect Machine Check Error Reporting Enable Correctly

**Problem:** IA32_MC1_STATUS MSR (405H) bit[60] (EN- Error Enabled) is supposed to indicate whether the enable bit in the IA32_MC1_CTL MSR (404H) was set at the time of the last update to the IA32_MC1_STATUS MSR. Due to this erratum, IA32_MC1_STATUS MSR bit [60] instead reports the current value of the IA32_MC1_CTL MSR enable bit.

**Implication:** IA32_MC1_STATUS MSR bit [60] may not reflect the correct state of the enable bit in the IA32_MC1_CTL MSR at the time of the last update.

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG25 Split Locked Stores or Locked Stores Through Certain Segments May not Trigger the Monitoring Hardware

**Problem:** Logical processors normally resume program execution following the MWAIT, when another logical processor performs a write access to a WB cacheable address within the address range used to perform the MONITOR operation. Due to this erratum, a logical processor may not resume execution until the next targeted interrupt event or O/S timer tick following a locked store within the monitored address range that either spans across cache lines or uses a segment register whose segment base is non-cacheline aligned.

**Implication:** The logical processor that executed the MWAIT instruction may not resume execution until the next targeted interrupt event or O/S timer tick in the case where the monitored address is written by a locked store which is either split across cache lines or through a segment whose segment base bits 5 to 0 are non-zero.

**Workaround:** Avoid accessing the monitored address range using either locked stores that split cache lines or locked stores that use a segment with a non-cacheline aligned segment base. It is possible for the BIOS to contain a workaround for this erratum

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG26 When BIST is Enabled, Warm Reset Incorrectly Clears IA32_FEATURE_CONTROL MSR and the Last Exception Record MSRs

**Problem:** IA32_FEATURE_CONTROL MSR (3AH), MSR_LER_FROM_LIP MSR (1DDH), and MSR_LER_TO_LIP MSR (1DEH) are cleared during warm reset when BIST (Built-In Self Test) is enabled. These MSRs should only be cleared on a power-up reset and not on a warm reset. A warm reset is different from a power-up reset in that PWRGOOD remains active throughout the assertion of RESET#.

**Implication:** Due to this erratum, any warm reset will clear IA32_FEATURE_CONTROL MSR, MSR_LER_FROM_LIP MSR, and MSR_LER_TO_LIP MSR content when BIST is enabled.

**Workaround:** BIOS or other firmware software must save IA32_FEATURE_CONTROL MSR, MSR_LER_FROM_LIP MSR, and MSR_LER_TO_LIP MSR information before warm reset and restore and reprogram the MSRs after the warm reset.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG27 CPUID Instruction Returns Incorrect Brand String

**Problem:** When a CPUID instruction is executed with EAX = 80000002H, 80000003H and 80000004H on an Intel Atom processor, the return value contains the brand string Intel(R) Core(TM) CPU when it should have Intel(R) Atom(TM) CPU.

**Implication:** When this erratum occurs, the processor will report the incorrect brand string.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG28 The Instruction Cache Does Not Respond to Snoops When All Logical Processors on a Core Are in an Inactive State

**Problem:** When all logical processors on a core enter an inactive state (for example MWAIT or HLT), the processor may incorrectly stop flushing lines in its instruction cache in response to snoops. This may cause the processor to not detect that memory has been modified and to execute the old instructions after waking up instead of the new contents of memory.

**Implication:** The processor may execute incorrect instructions after waking up from an inactive state.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG29 LINT0 Assertion and Deassertion During an Inactive State May Cause Unexpected Operation When APIC is Disabled

**Problem:** An interrupt delivered via LINT0 pins when the APIC is hardware disabled (IA32_APIC_BASE MSR (1BH) bit [11] is cleared) will usually keep the pin asserted until after the interrupt is acknowledged. However, if LINT0 is asserted and then deasserted before the interrupt is acknowledged and both of the following are true;

- The APIC is hardware disabled (IA32_APIC_BASE MSR bit [11] is clear) and
- The processor is in an inactive state that was requested by MWAIT, I/O redirection, VM-entry or RSM, then the processor may operate incorrectly.

**Implication:** Due to this erratum, the processor may run unexpected code and/or generate an unexpected exception. Intel has not observed this erratum with any commercially available software.

**Workaround:** If LINT0 is used, it is recommended to either leave the APIC enabled (IA32_APIC_BASE MSR bit [11] set to 1) or do not use MWAIT, I/O redirection, VM-entry or RSM to enter an inactive state.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG30 Processor May Not Wake-up from an Inactive State When an Enhanced Intel® SpeedStep Technology Transition is Pending

**Problem:** Due to this erratum, the processor may hang in rare scenarios when it is in inactive state and there is an Enhanced Intel SpeedStep® Technology transition pending.

**Implication:** The processor may hang and will be unable to resume execution. A processor reset will be needed to restart processor execution. Intel has not observed this erratum with any commercially available software.

**Workaround:** It is possible for the BIOS to contain a workaround to this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG31 Thermal Interrupts are Dropped During and While Exiting Intel® Deep Power-Down State

**Problem:** Thermal interrupts are ignored while the processor is in Intel Deep Power Down Technology state as well as during a small window of time while exiting from Intel Deep Power Down Technology state. During this window, if the PROCHOT signal is driven or the internal value of the sensor reaches the programmed thermal trip point, then the associated thermal interrupt may be lost.

**Implication:** In the event of a thermal event while a processor is waking up from Intel Deep Power-Down State, the processor will initiate an appropriate throttle response. However, the associate thermal interrupt generated may be lost.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG32 Corruption of CS Segment Register During RSM While Transitioning From Real Mode to Protected Mode

**Problem:** During the transition from real mode to protected mode, if an SMI (System Management Interrupt) occurs between the MOV to CR0 that sets PE (Protection Enable, bit 0) and the first far JMP, the subsequent RSM (Resume from System Management Mode) may cause the lower two bits of CS segment register to be corrupted.

**Implication:** The corruption of the bottom two bits of the CS segment register will have no impact unless software explicitly examines the CS segment register between enabling protected mode and the first far JMP. *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*, in the section titled "Switching to Protected Mode" recommends the far JMP immediately follows the write to CR0 to enable protected mode. Intel has not observed this erratum with any commercially available software.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG33 TSC May be a Lower Value After Being in the Deeper Sleep State

**Problem:** Due to this erratum, the TSC (Time-Stamp-Counter) may be observed to be a lower value after being in the Deeper Sleep State, irrespective of Deeper Sleep State duration.

**Implication:** Software may read a lower TSC value after exiting the Deeper Sleep State than it read before entering the Deeper Sleep State.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG34 GP and Fixed Performance Monitoring Counters With AnyThread Bit Set May Not Accurately Count Only OS or Only USR Events

**Problem:** A fixed or GP (general purpose) performance counter with the AnyThread bit (IA32_FIXED_CTR_CTRL_MSR (38DH) bit[2] for IA32_FIXED_CTR0, bit[6] for IA32_FIXED_CTR1, bit [10] for IA32_FIXED_CTR2; IA32_PERFEVTSEL0 MSR (186H)/ IA32_PERFEVTSEL1 MSR (187H) bit [21]) set may not count correctly when counting only OS (ring 0) events or only USR (ring>0) events. The counters will count correctly if they are counting both OS and USR events or if the AnyThread bit is clear.

**Implication:** A performance monitor counter may be incorrect when it is counting for all logical processors on that core and not counting at all privilege levels. This erratum will only occur on processors supporting multiple logical processors per core.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG35 CPUID Instruction Returns Incorrect Value For Leaf 0xA

**Problem:** When a CPUID instruction is executed with EAX = 0AH, the value returned in EDX is 0x2501, which reports support for only one fixed-function performance counter and also has an undefined bit [bit 13] set. The value of EDX should be 0x0503, reflecting that three fixed-function performance counters are supported.

**Implication:** When this erratum occurs, the processor will report an incorrect value in EDX.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG36 PMI Request is Not Generated on a Counter Overflow if its OVF Bit is Already Set in IA32_PERF_GLOBAL_STATUS

**Problem:** If a performance counter overflows and software does not clear the corresponding OVF (overflow) bit in IA32_PERF_GLOBAL_STATUS MSR (38Eh) then future overflows of that counter will not trigger PMI (Performance Monitoring Interrupt) requests.

**Implication:** If software does not clear the OVF bit corresponding to a performance counter then future counter overflows may not cause PMI requests.

**Workaround:** Software should clear the IA32_PERF_GLOBAL_STATUS.OVF bit in the PMI handler.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG37 CPUID Indicates Wrong L2 Associativity in Leaf 80000006H

**Problem:** When a CPUID instruction is executed with EAX=80000006H on a processor with a 512K L2 cache, it incorrectly returns 08H in ECX[15:12] which indicates a 16-way set associative L2. The return value in ECX[15:12] should have been 06H to indicate a 8-way set associative L2.

**Implication:** CPUID will report the L2 set associativity as 16-way when it should report 8-way.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG38 Code Fetch May Occur to Incorrect Address After a Large Page is Split Into 4-Kbyte Pages

**Problem:** If software clears the PS (page size) bit in a present PDE (page directory entry), that will cause linear addresses mapped through this PDE to use 4-KByte pages instead of using a large page after old TLB entries are invalidated. Due to this erratum, if a code fetch uses this PDE before the TLB entry for the large page is invalidated then it may fetch from a different physical address than specified by either the old large page translation or the new 4-KByte page translation. This erratum may also cause speculative code fetches from incorrect addresses.

**Implication:** The processor may fetch code from an incorrect address after a large page is converted into 4-KByte pages.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG39 Processor May Contain Incorrect Data and Hang Upon a Snoop When Combined with Specific Other Internal Conditions

**Problem:** In a specific corner case a snoop to a processor may cause incorrect data that will be followed by a hang.

**Implication:** Due to this erratum, the processor may contain incorrect data and hang in this specific circumstance.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG40 Complex Conditions Associated With Instruction Page Remapping or Self/Cross-Modifying Code Execution May Lead to Unpredictable System Behavior

**Problem:** Under a complex set of internal conditions, instruction page remapping, or self/cross modifying code events may lead to unpredictable system behavior

**Implication:** Due to this Erratum, unpredictable system behavior may be observed. ~~Intel has not observed this erratum with any commercially available software.~~

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG41 REP MOVS/STOS Executing With Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations

**Problem:** Under the conditions described in the Software Developers Manual section “Fast String Operation,” the processor performs REP MOVS or REP STOS as fast strings. Due to this erratum, fast string REP MOVS/REP STOS instructions that cross page boundaries from WB/WC memory types to UC/WP/WT memory types, may start using an incorrect data size or may observe memory ordering violations.

**Implication:** Upon crossing the page boundary, the following may occur, dependent on the new page memory type:

- UC: The data size of each read and write may be different than the original data size.
- WP: The data size of each read and write may be different than the original data size and there may be a memory ordering violation.
- WT: There may be a memory ordering violation.

**Workaround:** Software should avoid crossing page boundaries from WB or WC memory type to UC, WP or WT memory type within a single REP MOVS or REP STOS instruction that will execute with fast strings enabled.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAG42 Paging Structure Entry May be Used Before Accessed And Dirty Flags Are Updated

**Problem:** If software modifies a paging structure entry while the processor is using the entry for linear address translation, the processor may erroneously use the old value of the entry to form a translation in a TLB (or an entry in a paging structure cache) and then update the entry’s new value to set the accessed flag or dirty flag. This will occur only if both the old and new values of the entry result in valid translations.

**Implication:** Incorrect behavior may occur with algorithms that atomically check that the accessed flag or the dirty flag of a paging structure entry is clear and modify other parts of that paging structure entry in a manner that results in a different valid translation.

**Workaround:** Affected algorithms must ensure that appropriate TLB invalidation is done before assuming that future accesses do not use translations based on the old value of the paging structure entry.

**Status:** For the steppings affected, see the Summary Tables of Changes.

§

# Specification Changes

## AAG1: Implementation of System Management Range Registers

This processor has implemented SMRRs (System Management Range Registers). SMRRs are defined in Section 10.11.2.4 of the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide.

SMM (System Management Mode) code and data reside in SMRAM. The SMRR interface is an enhancement in Intel® 64 and IA-32 Architectures to limit cacheable reference of addresses in SMRAM to code running in SMM. The SMRR interface can be configured only by code running in SMM.

Under certain circumstances, an attacker who has gained administrative privileges, such as ring 0 privileges in a traditional operating system, may be able to reconfigure an Intel processor to gain access to SMM. The implementation of SMRR mitigates this issue. Intel has provided a recommended update to system and BIOS vendors to incorporate into their BIOS to resolve this issue.

§

# Specification Clarifications

There are no specification clarifications in this revision of the specification update.

§

# Documentation Changes

There are no documentation changes in this revision of the specification update.

***Note:*** Documentation changes for Intel® 64 and IA-32 Architecture Software Developer's Manual volumes 1, 2A, 2B, 3A, and 3B will be posted in a separate document, Intel® 64 and IA-32 Architecture Software Developer's Manual Documentation Changes. Follow the link below to become familiar with this file.

http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

§